RESOURCE-USAGE NOTIFICATION FRAMEWORK IN A DISTRIBUTED COMPUTING ENVIRONMENT

TECHNICAL FIELD

[0001] The present disclosure relates generally to resource usage in distributed computing systems. More specifically, but not by way of limitation, this disclosure relates to a resource-usage notification framework in a distributed computing environment.

BACKGROUND

[0002] There are various types of distributed computing systems, such as grid computing systems, computing clusters, and cloud computing systems. Among these, computing clusters and cloud computing environments have become increasingly popular.

[0003] A computing cluster (cluster) is a group of nodes that can work together like a single system to combine computing power. Each node in the cluster may be running the same operating system and have other commonalities. A cluster can assign the same computing task to every node, or can assign different computing tasks to different nodes.

[0004] Cloud computing systems have a shared pool of computing resources (e.g., servers, storage, and virtual machines) that are used to provide services to users on demand. These services are generally provided according to a variety of service models, such as Infrastructure as a Service (Iaas), Platform as a Service (PaaS), or Software as a Service (SaaS). But regardless of the service model, cloud providers manage the physical infrastructures of the cloud computing systems to relieve this burden from users, so that the users can focus on other tasks like deploying applications.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. **1** is a block diagram of an example of a distributed computing environment for implementing a resource-usage notification framework according to some aspects.

[0006] FIG. **2** is a block diagram of an example of a system for implementing a resource-usage notification framework according to some aspects.

[0007] FIG. **3** is a flow chart of an example of a process for implementing a resource-usage notification framework in a distributed computing environment according to some aspects.

DETAILED DESCRIPTION

[0008] There are various types of software applications, such as monolithic software applications and distributed software applications (e.g., software applications running on multiple nodes in a distributed computing system). Such software applications often use variable and unknown amounts of resources to perform tasks. Examples of such resource usage can include memory usage, disk usage, network usage, and processing-unit usage. If a software application overuses one or more resources, a cluster typically kills (e.g., terminates) the software application to preserve resources for other software applications running in the cluster. After a software application is killed, it may need to start over. In some cases, software applications may not save their data before they are killed, particularly if the software application is stateless. Rerunning such software applications, particularly when they are being used for big-data processing and machine learning, can be computationally expensive if they are killed in the middle of processing due to reaching resource limits.

[0009] Some examples of the present disclosure can overcome one or more of the abovementioned problems by notifying software applications that they are approaching a resource-usage limit so that the software applications can take corrective action, for example, to prevent the software application from being killed. In one example, a system can determine the resource usage of a software application in a distributed computing environment. The system can determine if the resource usage is within a predefined range of a predefined resource-consumption limit that can indicate the resource is overused. If so, the system can generate an event notification and transmit the event notification to the software application. The software application can receive the event notification and perform a mitigation operation in response. The mitigation operation can be configured to prevent the resource usage from exceeding the predefined resource-consumption limit or to mitigate an impact of the resource usage exceeding the predefined resource-consumption limit.

[0010] In one particular example, a software application can be a stateless application running on one or more nodes of a distributed computing environment (e.g. a cloud computing environment). A stateless application can be a software application that does not maintain data between sessions. The distributed computing environment can be configured to automatically stop or shutdown the software application in response to the software application's resource usage exceeding a predefined resource-consumption limit. To prevent this from happening, some examples can include a monitoring agent running on the one or more nodes associated with the software application. The monitoring agent can receive information about resource usage (e.g., memory usage, network usage, disk usage, or processing-unit usage) by the software application. The resource usage information could be received from an operating system kernel in one example. The monitoring agent can transmit the resource usage information to a notification controller running within the cluster. The notification controller can determine if the resource usage approaches the predefined resource-consumption limit and, if so, generate an event notification. The notification controller can then transmit the event notification to the software application. In response to receiving the event notification, the software application can perform a mitigation operation.

[0011] The mitigation operation can include one or more predefined operations configured to prevent the resource usage from exceeding the predefined resource-consumption limit or to mitigate an impact of the resource usage exceeding the predefined resource-consumption limit. In some examples, the software application can access a lookup table that specifies the mitigation operation for a given event notification.

[0012] As one particular example, the lookup table can specify that, for a given event notification, the intermediate results of a data-processing operation implemented by the software application are to be stored. Storing the intermediate results of the data-processing operation can enable the software application to pick up the data-processing operation from where it left off when the software application was